Integration of NaradaBrokering and Audio/Video Conferencing as a Web Service

Hasan Bulut[1,2], Geoffrey Fox[1,2,3], Shrideep Pallickara[1],Ahmet Uyar[4] and Wenjun Wu[1]
[1]Community Grid Computing Laboratory, Indiana University
hbulut@indiana.edu, **gcf@indiana.edu**, **spallick@indiana.edu**, **wewu@indiana.edu**
[2]Department of Computer Science, Indiana University
[3]School of Informatics and Physics Department, Indiana University
[4]Department of Electrical Engineering & Computer Science, Syracuse University
auyar@syr.edu

Abstract
Audio/Video Conferencing Systems need communication channels between their clients in order to transport RTP packets from one client to another. In this paper we investigate audio/video conferencing as a Web Service and the deployment of publish/subscribe systems in the context of audio/video conferencing systems. In this paper we use our research system NaradaBrokering, which supports both peer-to-peer and publish/subscribe paradigms, as a test bed to investigate these ideas. We also present results from our research system.

Keywords: audio/video (A/V) conferencing, web services, publish/subscribe systems, middleware, RTP

1. Introduction
The most common way of transmitting multimedia (audio and video) traffic on the Internet is to use RTP [22] (A Transport Protocol for Real-Time Applications). Although RTP is independent of the transport layer, the most commonly used transport mechanisms are UDP and multicast. TCP is also used occasionally, especially when transferring media behind a firewall, since UDP or multicast cannot go through firewalls in general. In this paper we investigate transporting RTP traffic using the publish-subscribe messaging paradigm. Though this calls for extra latency and bandwidth requirements, we believe there are several benefits to be accrued by such integrated solutions.

RTP provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio and video. When sending media packages over Internet, an RTP header is added to each package. This RTP header is usually 12 bytes and contains information about the media it is carrying such as the media *codec* type, timestamp (sampling time of that package) and source identifier. All the packages belong to the same media stream have the same source identifier therefore the receiver can reconstruct the stream from independent RTP packages. RTP also figures out the correct sequence of the packages from the timestamps of packages and sorts them to get the original sequence of media stream. RTP uses RTCP (RTP Control Protocol) to monitor the timely delivery of real-time data. It provides control and identification functionality. RTCP packages are very similar to RTP packages but they do not contain any media information, rather they contain information about the identity of the sender and the quality of media transfer. RTCP packages are usually sent every 3 seconds.

Multimedia applications mandate timely delivery of content and generally sustain loss of media packets very well. This makes UDP a very good choice for transporting media, since unlike TCP it does not incorporate an error detection/correction mechanism, which add to delays associated with individual packets. Since UDP is point to point, it is best to use RTP over UDP when sending streams from one client to another directly. RTP over UDP does not provide any support for group communication. To hold online audio or video meetings using RTP over UDP, one should have a media server which gets the media streams from senders in a meeting and distributes it to the receivers while duplicating the streams whenever necessary.

In the case of using multicast to transport RTP packets, it is the network's responsibility to deliver content to the interested recipients. Users send the RTP packages to an agreed upon virtual multicast address and interested parties receive content by registering an interest to the multicast address. Deployment of multicast, though it is a connectionless delivery mechanism similar to UDP, for dissemination of content, has different characteristics than that of using UDP. In multicast the network is responsible for duplicating the streams when necessary, without bothering the sender prior to delivery to interested parties. The routing of media to recipients is thus delegated to routers. It is very easy to setup audio/video conferencing in a multicast environment. Simply agree on a multicast ip address and port number, and you can exchange media with other participants. There is also no need for dedicated media server to transport content.

There are three significant issues that can curtail the efficient deployment of multicast based multimedia solutions. First, in multicast there is no one authority that assigns Multicast-address/port number pairs to interested users. Furthermore, there is no way of limiting the use of a multicast-address/port number pair since anyone can use any multicast-address/port number of the multicast domain at anytime. It is thus possible to eavesdrop on meetings in

session without detection. Besides such access to media anyone can launch a denial of service attack by sending voluminous media streams to multicast address. Although, it is possible to encrypt the RTP messages so that unwanted users can not access the media, currently there is no way of avoiding the denial of service attacks. Secondly, multicast requires support at the router level. Although most routers have support for multicast, it is usually disabled. Most organizations disallow multicast traffic since it can be very bandwidth intensive causing applications inside an organizational domain to suffer. Thirdly, multicast IP domain is static and limited. If there are more people who want to use multicast frequently multicast IP-address/port number collisions may occur often. It is also imperative that the TTL associated with the datagram packets in multicast needs to be used prudently to ensure good attenuation of the multicast traffic.

2. RTP over Messaging Systems

We suggest that transferring media over the Internet using the publish/subscribe messaging model have several benefits. This approach would of course be similar to multicast in the sense that a distributed network of brokers will handle the routing of media as opposed to routers. This scheme would also posses an ease of use similar to that of multicast – to create an online meeting, a user only needs to create a topic on the broker. But this system will not have the disadvantages of multicast.

Currently messaging systems based on the Java Message Service (JMS) [15] publish/subscribe specification are mostly used for asynchronous delivery of reliable data over TCP. We propose the use of JMS style systems for the unreliable delivery of multimedia data over UDP. In this paper we are going to represent the results of transporting media data using our JMS compliant brokering system, NaradaBrokering.

In the context of a video conferencing application when a video stream is fed to a topic, any number of clients can receive it simply by subscribing to the topic to which the stream is being published. The broker network handles the software multicast of streams to relevant subscribers. The software multicast approach ensures that the solution can be deployed and will work anywhere. Utilization of network resources can also be very efficient in such systems. One problem in multicast sessions is that any user can send streams to the virtual multicast address in use for a given meeting. In the case of messaging systems this can be prevented easily in the context of publish/subscribe brokering since it is very easy to control the subscribers and the publishers to the topics. The brokers can easily incorporate an authentication scheme besides efficiently controlling the subscription and publishing rights among authenticated users.

Publish-subscribe systems can also be used for other networking applications. In fact our Garnet [21] collaboration environment is based on JMS and provides application sharing, whiteboard, chat etc. Publish-subscribe brokering can thus provide a unified framework for a wide gamut of applications from media communications to application sharing. Such an approach simplifies the management of applications considerably.

A disadvantage of this scheme is that we carry additional header information on media packets. This header is necessary for routing messages within the messaging system. Since the approach entails extra header information within each media package, there is an increase in the amount of data transferred over the network, which in turn results in the need for more bandwidth. In JMS based messaging, the header could be around 200 bytes compared to 12 bytes of RTP header that accompany the data packets encapsulating audio and video data. In NaradaBrokering we could define a special data type for audio/video content where the headers are significantly smaller in size. Also, since the routing will be done in software, it will be slower than multicast but our results have indicated that the performance (2-4 milliseconds) is sufficient for most applications. Moreover, with CPU performance and network bandwidth increasing rapidly, the flexibility offered by our approach offsets the small loss in performance. In [21] we have shown that latencies in the order of a few milliseconds are sufficient for most collaborative applications.

3. NaradaBrokering

NaradaBrokering [3-7] is an event brokering system designed to run on a large network of cooperating broker nodes. Communication within NaradaBrokering is asynchronous and the system can be used to support different interactions by encapsulating them in specialized events. The NaradaBrokering scheme of automating broker additions within a distributed cluster architecture, while resulting in the creation of “small world networks” [1,2], allows support for large heterogeneous client configurations that scale to arbitrary size. NaradaBrokering guarantees delivery of events in the presence of failures and prolonged client disconnects, and ensures fast dissemination of events within the system. To ensure fast dissemination of events within the system, NaradaBrokering relies on broker network maps hosted at each broker to compute the fastest routes to reach a targeted set of destinations. The routing in NaradaBrokering is very efficient since each broker computes shortest paths to reach a set of destinations, and the only brokers that are involved in the route calculations

are those that have not failed or have not been failure suspected.

NaradaBrokering is JMS compliant and provides support not only for JMS clients, but also for replacing [16] single/limited server JMS systems transparently with a distributed Narada broker network. Since JMS clients are vendor agnostic, this JMS integration has provided Narada with access to a plethora of applications built around JMS, while the integrated Narada-JMS solution provides these applications with scaling, availability and dynamic real time load balancing. Among the applications ported to this solution is the Anabas distance education [18] conferencing system and the Online Knowledge Center (OKC) portal [19] being developed at the IU Grid labs. NaradaBrokering currently also incorporates a UDP transport based JMS-style solution that could be used to disseminate transient JMS messages. NaradaBrokering also provides support for peer-to-peer (P2P) communications [9-11] by providing support for JXTA (from *juxta*position) [12,13] interactions. Additional information pertaining to the integration of JXTA and NaradaBrokering can be found in [17].

4. A/V Collaboration Web Service

The A/V Collaboration Web Service [14] developed by us is an audio-video conferencing system in which the Session Server, H.323 [25] Gateway, SIP [24] Gateway and Media Server Gateway are implemented as standalone web services. The messaging format, XML-Based General Session Protocol (XGSP), used for communication between these services is as the name suggests XML based. XGSP enables WSDL-based [23] collaborating clients to create dynamic groups and join these groups to share various collaborative multimedia streams such as audio, video. There are basically four sets of methods in XGSP: Registration Method, Session Command, Session Channel Binding and Query Method. The Registration Method allows users to register themselves in a registration server. Session Command is divided in two groups: one for the session control, such as Source Select Request, Request/Release Chairman, Request/Release/ Grant/Cancel Floor and the other group is for the membership of the session, such as Create Session, Join Session, Leave Session and so on. Session Channel Binding Method is used to bind the RTP channels of a client into the media server. Finally, using Query Method, clients and the session server can discover various properties of the system, such as how many sessions are going on.

5. Integrating NaradaBrokering and A/V Collaboration

In order to integrate the client with Narada and with the A/V Collaboration Web Service, we need two adapters. The Web Service Adapter will be used to communicate with the A/V Collaboration Web Service and the Narada Adapter will be used to communicate with Narada. One can suggest that these adapters may be placed outside the client computer, but because of the firewall problems and performance issues we currently recommend that these adapters are placed within the client computer. Figure 1 shows how this integration be achieved.

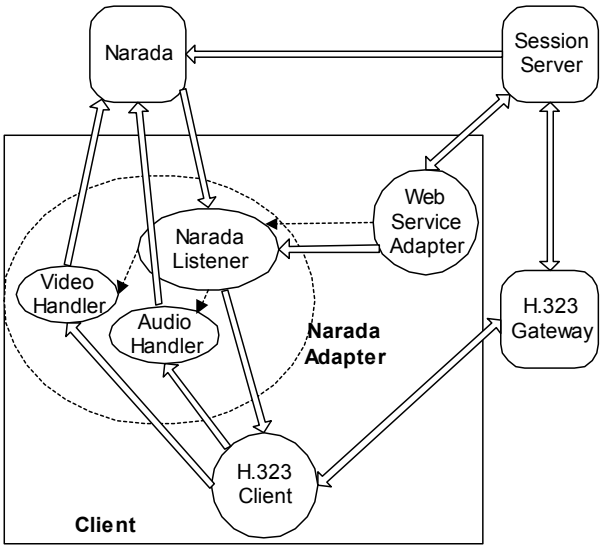

Figure 1: Integration of Narada and A/V Web Service

5.1 XGSP Additions

For the purpose of the integration, we have added new message formats, which will be used between Web Service Adapter and A/V Collaboration Web Service. Tables 2 and 3 outline the fields (and their functions) in the newly added XML encoded XGSP messages.

Field	Explanation
Client ID	A unique name for the client
IP Address	The IP address of the client

Table 1: Join Request from client to A/V Collaboration Web Service

Field	Explanation
Client ID	The unique name for the client
Narada System Info	Server name, port number, topic name
Listener	Port numbers for audio and video, to where the RTP packets will be sent
Publisher	Port numbers for audio and video, from where the RTP packets will be received

Table 2: Join-Reply message

5.2 Web Service Adapter
A web service adapter starts the communication with the A/V Web Service in order to make the first request to join to the A/V session and receive the Narada parameters for the Narada Adapter. As described above in tables 1 and 2, XGSP protocol is updated so that the web service would be able to define and send the Narada parameters to the Web Service Adapter.

5.3 Narada Adapter
Web Service Adapter generates Narada Adapter to establish the communication between the client conferencing tool and the Narada system. The components of the Narada Adapter are Narada Listener, Audio Handler and Video Handler. Narada Listener subscribes to the Narada System to receive the messages from the specified topics for audio and video messages. Then, it extracts the RTP packets from these messages and forwards them to the appropriate port of the client conferencing tool, such as an H.323 client. Audio Handler and Video Handler receive RTP packets from the client. Then, these handlers publish the audio and video packets received from the H.323 client to Narada.

5.4 Communication between Client-Adapters & A/V Web Service
Initially the Web Service Adapter sends join messages to the Session Server. The H.323 client then initiates connection with H.323 Gateway. This negotiation takes place according to the H.323 protocol. Meanwhile, Session Server receives the necessary information from the H.323 Gateway and sends a join-reply message to the Web Service Adapter, including the NaradaBrokering session parameters, audio and video port numbers to send and receive the streams. Web Service Adapter generates the Narada Adapter, whose components are Narada Listener, Audio Handler and Video Handler. H.323 Client is told to send the RTP packets to local ports as the destination address and ports. H.323 client would receive the streams as soon as the Narada Listener receives messages from Narada system. While leaving the session, simply H.323 client sends a BYE message to the H.323 Gateway and session server sends another message to the Web Service Adapter to destroy the listeners and publishers and end the session.

6. Performance

We compare the performance of the transfer of RTP packets using the Java Media Framework and NaradaBrokering's UDP based JMS solution. In this experiment we use two Red Hat 7.3 linux machines, one for clients and one for servers. The machine hosting the clients is a 1.80GHz Intel Pentium 4 with 512MB of memory. The server machine is a 1.266GHz dual CPU Intel Pentium 3 with 1024 MB of memory. The experiments and the results that we have included in this section are our preliminary results. We intend to continue gathering performance numbers for the final version of this paper.

The client machine runs the transmitter and the receiver clients. We host both the transmitter and receiver on the same machine to obviate the need for clock synchronizations and the need to account for clock drifts, while computing the delay in the delivery of individual packets. The second machine runs a reflector server and a NaradaBrokering broker. All processes involved in the experimental setup use the Blackdown-1.3.1, Java 2 JRE JVM. The transmitter client is a Java program written using Java Media framework API. It reads a media file and sends it over the network. The receiver client is also a Java program written using the JMF API. It gets a media stream over the network and plays it. The reflector server is also a program written using JMF API. It receives a media stream from the network and sends it to another IP address. Our benchmark uses an H.263 video file that is a 30 second part of a movie, with an average bit-rate of 600Kbps (Kilo bits per second) and a frame-rate of 30 frames/sec. The transmitter client reads this file from the disk and sends it to the server machine. Then reflector server or the NaradaBrokering broker sends it back to the receiver client, which plays it.

The reflector case corresponds to using UDP to transfer RTP packages, while the NaradaBrokering case corresponds to transferring RTP packets inside JMS messages and transferring them using UDP. For every packet that is received we compute the transit delay associated with the delivery of RTP packets. We also measure the Jitter J, which is defined by the RTP RFC as the mean deviation (smoothed absolute value) of the difference D in packet spacing at the receiver compared to the sender for a pair of packets. The Jitter J is computed based on the formula –
$J = J + (|D(i-1, i)| - J)/16$, where $D(i-1, i)$ corresponds to the difference between the delay for i^{th} RTP packet and the delay for the $(i-1)^{th}$ RTP packet. For the sample of packets that are received we also compute the mean delay and the standard deviation associated with the delays for individual packets. For computing the standard deviation in both cases we ignore the first 60 delays samples since they correspond to the start up of the application. Figures 2 and 3 depict the delays and jitter (up until that point) values associated with individual packets.

The delays and jitters at start of the video session are high because of the JMF player initialization. In both cases individual packets arrive on time but are not being processed in a timely fashion by the player. We ignore these first few samples that correspond to initialization

latencies while computing the standard deviation and mean delay for both cases.

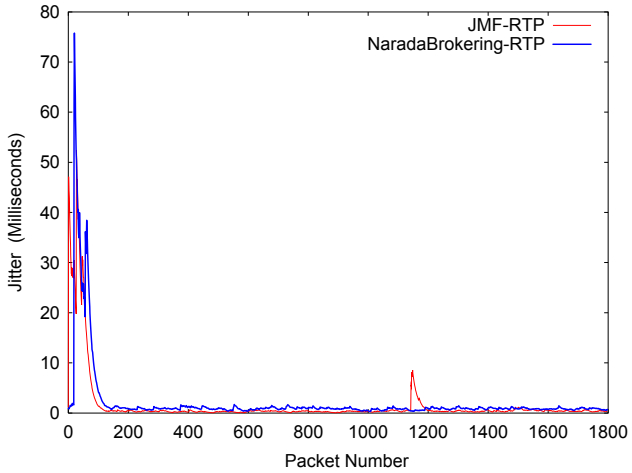

Figure 2: Comparison of Jitter in NaradaBrokering and JMF

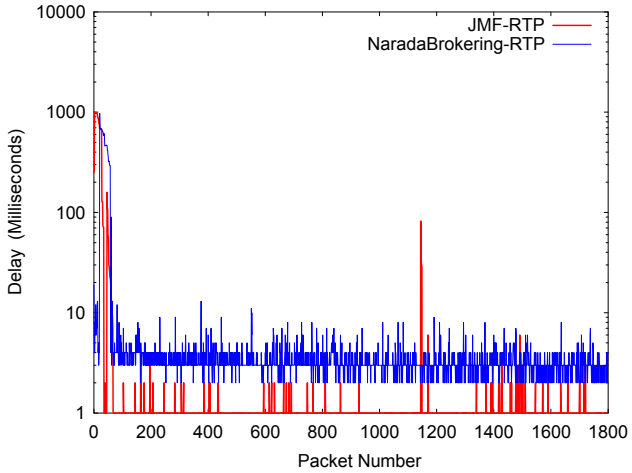

Figure 3: Comparison of delays for individual packets in NaradaBrokering and JMF

As can be seen in figure 2 the jitter in Narada is very similar to the JMF-RTP case with the exception of the spike that exists in the JMF-RTP case. The results, in figure 3, for delay indicate that the JMF-RTP delays are better than those for NaradaBrokering. However the delays in Narada are still in the range (mean=3.5millisecond) which would facilitate development of sophisticated A/V conferencing environments. The overheads associated with the marshalling and unmarshalling of JMS packets along with the additional JMS headers, associated with every JMS message, add to the delay associated with individual packets in the NaradaBrokering-RTP-JMS case. The delay associated with routing individual packets while using NaradaBrokering would be significantly reduced under two conditions. First, as the geographic distances between the transmitter and receiver increase the affects of marshalling/unmarshalling and increase in packet size will not predominate the delay (communications across 1000 miles generally tend to be in the range of 10 milliseconds in which case the curves for JMF-RTP and NaradaBrokering-JMS-RTP would be very close). Second, we could construct a special Narada message for handling audio/video packets with a significantly lower header size. Furthermore, we do understand how a production version of the NaradaBrokering system could give significantly better performance – about a factor of 2-3 lower in latency than the current research prototype. By improving the thread scheduling algorithms and incorporating flow control (needed at high publish rates) into the NaradaBrokering core significant gains in performance can be achieved.

Table 3: Comparing NaradaBrokering-RTP (JMS) and JMF-RTP

	Mean Delay (milliseconds)	Standard Deviation (milliseconds)
JMF-RTP	0.898	0.494
NaradaBrokering JMS-RTP	3.282	0.877

Table 3 provides the mean delay and standard deviation associated with our runs. These results demonstrate that NaradaBrokering-JMS-RTP case has a comparable performance to the JMF-RTP. Also note that in NaradaBrokering we can support heterogeneous transport protocols; NaradaBrokering can easily switch (like JXTA based systems) between UDP and TCP/IP for different hops.

7. Comparison with VRVS

VRVS (Virtual Rooms Videoconferencing System) [20] is a web-oriented system for videoconferencing and collaborative work over IP networks. Using VRVS, users from disparate geographic locations can meet and participate in MBONE, H.323 or MPEG2 multipoint videoconferences. Participants can make use of different collaborative tools (sharing their desktops, broadcast any local application, participate in a chat, etc.). VRVS also integrates MBONE and H.323 tools with AccessGrid Virtual Venues. H.323 Clients are connected to the Virtual Rooms by the H.323–VRVS Gateway.

Reflectors play a big role in VRVS design. A reflector is a host that connects each client to a Virtual Room by a permanent IP tunnel. The reflectors have many IP tunnels among themselves. The clients multiplex these tunnels. The reflectors and their links form a set of virtual sub-networks through which audio, video or data flows. The use of the reflector technology assures the quality needed for videoconferences transmission. Reflectors are the ones responsible for transmitting the streams. The basic architecture of VRVS is shown in figure 4.

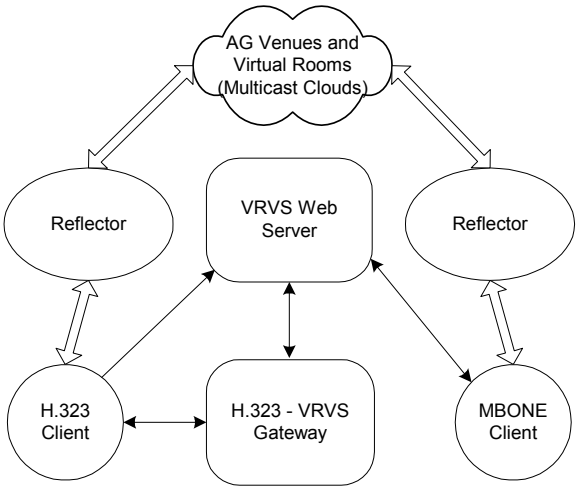

Figure 4: Basic Architecture of Virtual Rooms Videoconferencing Systems

The main difference between these two designs is, while VRVS uses unicast streaming, our design proposed in this paper uses event brokering systems in transmitting streams which ensures that only clients that are subscribed to a certain topic would receive the streams. In the event brokering system, the adapters allow using different client applications such as MBONE, H.323, and SIP. These adapters render the messages specific to the client application. In this design, different applications need different adapters due to the nature of the messages they accept. Finally, note that NaradaBrokering is a far more powerful than a reflector network since it fully incorporates the publish/subscribe paradigm.

8. Conclusion

In this paper, we have investigated the use of publish/subscribe systems in audio/video conferencing systems. We also discussed our strategy for integrating A/V as a Web Service. We also presented results from our preliminary investigations. We intend to augment these results with further experiments in the final version of this paper.

9. References

1. D.J. Watts and S.H. Strogatz. Collective Dynamics of Small-World Networks. *Nature*. 393:440. 1998.
2. R. Albert, H. Jeong and A. Barabasi. Diameter of the World Wide Web. *Nature* 401:130. 1999.
3. The NaradaBrokering System http://grids.ucs.indiana.edu/ptliupages/projects/narada/
4. Geoffrey Fox and Shrideep Pallickara, An Event Service to Support Grid Computational Environments, to be published in *Concurrency and Computation: Practice and Experience*, Special Issue on Grid Computing Environments.
5. Geoffrey C. Fox and Shrideep Pallickara, An Approach to High Performance Distributed Web Brokering. *ACM Ubiquity* Volume2 Issue 38. November 2001.
6. Pallickara, S., "A Grid Event Service." PhD Syracuse University, 2001.
7. Geoffrey C. Fox and Shrideep Pallickara .The Narada Event Brokering System: Overview and Extensions To appear in the Procedings of the *2002 International Conference on Parallel and Distributed Processing Techniques and Applications* (PDPTA'02).
8. Geoffrey Fox, Ozgur Balsoy, Shrideep Pallickara, Ahmet Uyar, Dennis Gannon, Aleksander Slominski. Community Grids. Proceedings of the *International Conference on Computational Science* (ICCS 2002). Amsterdam, Netherlands April 2002.
9. Geoffrey Fox, "Peer-to-Peer Networks," *Computing in Science & Engineering*, vol. 3, no. 3, May2001.
10. openp2p P2P Web Site from O'Reilly http://www.openp2p.com.
11. "Peer-To-Peer: Harnessing the Benefits of a Disruptive Technology", edited by Andy Oram, O'Reilly Press March 2001.
12. Sun Microsystems. The JXTA Project and Peer-to-Peer Technology http://www.jxta.org
13. The JXTA Protocol Specifications. http://spec.jxta.org/v1.0/docbook/JXTAProtocols.html
14. Geoffrey Fox, Wenjun Wu, Ahmet Uyar, and Hasan Bulut *A Web Services Framework for Collaboration and Audio/Videoconferencing*
15. Mark Happner, Rich Burridge and Rahul Sharma. Sun Microsystems. Java Message Service Specification. 2000. http://java.sun.com/products/jms
16. Geoffrey C. Fox and Shrideep Pallickara JMS Compliance in the Narada Event Brokering System. To appear in the proceedings of the *2002 International Conference on Internet Computing* (IC-02).
17. Geoffrey Fox, Shrideep Pallickara, Xi Rao and Qinglin Pei. A Scaleable Event Infrastructure for Peer-to-Peer Grids. *Under Review*.
18. The Anabas Conferencing System. http://www.anabas.com
19. The Online Knowledge Center (OKC) Web Portal http://judi.ucs.indiana.edu/okcportal/index.jsp
20. VRVS http://www.vrvs.org/
21. Geoffrey Fox et al. Grid Services For Earthquake Science. To appear in *Concurrency & Computation: Practice and Experience*. Special Issue on Grid Computing Environments.
22. RTP: A Transport Protocol for Real-Time Applications (IETF RFC 1889) http://www.ietf.org/rfc/rfc1889.txt.
23. Web Services Description Language (WSDL) 1.1 http://www.w3.org/TR/wsdl.
24. SIP: Session Initiation Protocol. IETF RFC 2543. http://www.ietf.org/rfc/rfc2543.txt
25. ITU-T H.323-Packet-Based Multimedia Communications Systems, Nov 2000